ELECTRON-CLOUD MODULE FOR THE ORBIT CODE†

A. Shishlo#, Y. Sato, J. Holmes, S. Danilov, S. Henderson,
SNS* project, ORNL, Oak Ridge, TN 37831, USA

Abstract

A new electron cloud module has been developed and inserted into the beam dynamics code, ORBIT, which is used for simulations in high intensity proton rings. In addition to incorporating the dynamics of electron cloud build up, the model simulates the self consistent dynamics of the proton beam and the electrons. This new model includes full 3D descriptions of the proton bunch and the electron cloud, including both their space charge interactions and their motion in external electric and magnetic fields. The secondary emission of electrons is calculated using a set of models based on those of M. Furman and M. Pivi. The structure, algorithms, parallel implementation, and benchmarks of the new ORBIT module with analytic results are presented.

MOTIVATION

The instability caused by an electron cloud effect (ECE) may set an upper limit to the intensity of proton storage rings. For instance, the instability observed in the Proton Storage Ring (PSR) at the Los Alamos National Laboratory is probably due to the interaction between electron clouds (EC) and the proton beam [1]. Similar instabilities have been observed in other machines, so that in spite of several prevention measures [2] ECE continues to be a concern in the Spallation Neutron Source accumulation ring. The self consistent description of the ECE including proton beam instabilities is a part of electron cloud dedicated studies at the SNS project.

SOLUTION

Any code simulating the collective coupling of the electron cloud and the proton beam must include three integrated parts: an accelerator code for proton beam dynamics calculations, a model for the electron cloud build up and dynamics, and an interface that contains the interactions between EC and protons.

Accelerator Code

There are two reasons that the ORBIT code [3] has been chosen as the accelerator component of a new EC module. First, from the beginning ORBIT was dedicated to the realistic simulation of actual machines, especially SNS. It includes many features to describe single particle and collective dynamics. Among these are: single particle transport through various types of lattice elements using either MAD matrices or simplectic TEAPOT-like elements; magnet errors; closed orbit calculations; orbit correction; longitudinal and transverse impedances; 1D, 2D, and 3D space charge models; losses and collimation; etc. These capabilities are important for distinguishing the effects of different possible sources of instabilities, including the electron cloud effect. Second, ORBIT supports parallel simulations using MPI. There is no doubt that the new EC module must support parallel calculations to achieve reasonable calculation times.

Electron Cloud Model

There are several simulation codes that describe the development and dynamics of electron clouds, but the effort and time required to integrate them with an existing accelerator code can be more than that needed to design and implement a new EC code based on well known algorithms and physical models. We therefore decided to develop a new code with following features:

- It is a collection of C++ classes.
- All components have a well defined interface.
- It can be easily extended by inheritance from base classes.
- It provides parallel calculations based on MPI.
- It can be used independently without a wrapping accelerator code.

Linkage between EC model and ORBIT

There are only a few classes needed to glue together the new ECE code and the original ORBIT code. They deal with the EC module and ORBIT classes as data members and orchestrate all computational flow.

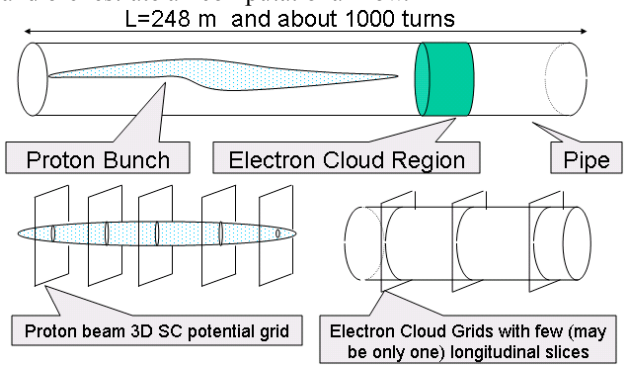

Figure 1: Simulated physical system.

PHYSICAL APPROACH

The simulated physical system consists of the proton bunch in the ring, electrons inside a special region that is called an electron cloud region, and a perfectly conducting pipe whose surface can be a source of primary or secondary electrons (see Fig. 1). The protons propagate

† Research sponsored by UT-Batelle, LLC, under contract no. DE-AC05-00OR22725 and DE-FG02-92ER40747 for the U.S. Department of Energy, and NSF under contract no. PHY-0244793.
# shishlo@ornl.gov
* SNS is a partnership of six national laboratories: Brookhaven, Argonne, Jefferson, Lawrence Berkeley, Los Alamos and Oak Ridge.

around the ring using ORBIT, with location *s* as independent variable, until they encounter an electron cloud region. At this time, they are frozen and passed through the electron cloud region where they contribute to the electron dynamics, which is calculated using time as the independent variable. The changes in proton momentum due to the electron cloud are accumulated as kicks in an auxiliary grid covering the proton beam and applied to protons at the end of propagation through the electron cloud region.

For both electrons and protons the PIC method is used to calculate fields, so subsidiary grids are needed for space charge densities and potentials. The proton bunch is very long (about 160 m in SNS) compared to its transverse size (a diameter of about 10 cm in SNS). Hence, the 3D grids are treated as a set of transverse 2D grids uniformly distributed along the longitudinal coordinate. For each 2D slice, an independent space charge problem is solved and this provides an opportunity for effective and simple parallelization of the code. This approach is applicable for long and thin bunches. These simplifications are completely reasonable for both SNS and PSR.

One can define multiple electron cloud regions in the ring lattice either to cover the most dangerous places with respect to ECE or, if desired, the whole ring. The length of each EC region should be short enough to provide small changes in twiss parameters inside. Each region has its own bunch of electrons with its own unique history and dynamics, and a set of external magnetic fields if we consider EC inside magnets. Interaction between the different electron clouds regions exists only through the proton beam dynamics.

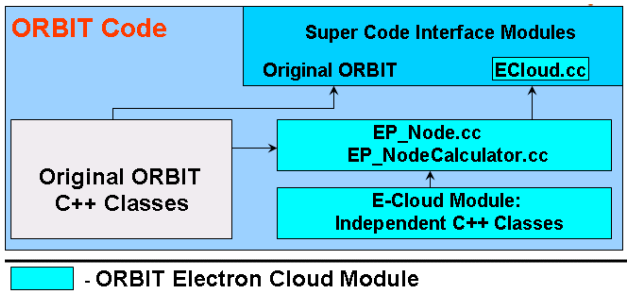

Figure 2: ORBIT's electron cloud module structure.

ELECTRON CLOUD MODULE STRUCTURE

The ORBIT code consists of a combination of C++ classes and SuperCode Interface modules. To use ORBIT, the user creates a SuperCode script that does not require compilation, and this script is interpreted by the ORBIT executable. One ongoing activity is the replacement of the SuperCode by a more powerful Python interface. The general structure of the EC module and its fit into the ORBIT code are shown in Fig. 2. The new module is mostly independent of the original ORBIT code and can be used either inside other accelerator codes or independently. The ECloud class is merely a SuperCode wrapper of the EP_Node and EP_NodeCalculator class methods. These classes interact with structures and classes of original ORBIT and organize the whole EC simulation process.

BASE CLASSES OF EC MODULE

There are only a few classes responsible for the functionality of the EC module. They are listed in Table 1. Others classes are for convenience and/or diagnostics classes and do not affect the new model simulation approach.

Table 1: Base classes of the EC module.

Class	Description
eBunch	Manipulates the 6D – coordinates of the macro-electrons.
EP_Boundary	2D SC solver that also contains the transverse 2D grid parameters.
Grid3D	3D grid with references to the EP_Boundary class.
Surfaces Classes	Collection of classes describing different surfaces of the beam pipe.
Field Source Classes	Collection of classes specifying electrostatic and magnetic fields.
Tracker	Tracks macro-electrons by using an arbitrary set of field sources.

eBunch Class

The eBunch class is a resizable container that keeps information about macro-electrons: 6D coordinates; a macro-size; a dead/alive flag. The macro-size of each macro-electron is a unique value that is defined at the moment of adding a new macro-electron to the bunch. This class provides the following methods to operate with macro-electrons: access to each of the 6D coordinates; add macro-electron; delete macro-electron; print all information into a file; create all macro-electrons by reading information from an external file. Its parallel capabilities are used when it reads and writes the content of the electron bunch into or from the external file.

EP_Boundary Class

The EP_Boundary class has mixed functionality:

- It keeps information about 2D transverse grid and beam-pipe shape and size in the XY-plane. The pipe shape can be a circle, ellipse, or rectangle.
- It contains a 2D Poisson solver that uses the convolution method. It contains a method that accepts a 3D grid with space charge density and returns another 3D grid with potential values at the grid points. Each XY-slice of the potential 3D grid is a solution of the 2D space charge problem for the XY-slice of the space-charge density grid.

- It can add boundary conditions (zero potential on the beam-pipe) to the potential 3D grid by using the Capacity Matrix Method.
- It uses an external FFTW library and keeps necessary arrays inside.
- For an electron hitting the surface of the beam-pipe, it finds an impact point on the surface and calculates the electron's normal vector by using internal geometry information
- It does not have any parallel capabilities

Grid3D Class

The Grid3D class is the parent class for a Grid3D class hierarchy that, in addition to the parent class, includes two subclasses that specifically describe a space charge density and potential grid.

The parent class has the following functions:

- It contains 3D arrays of charge density and potential values.
- It provides direct access to the 3D arrays and to the 2D slices.
- It calculates a charge density and gradient at an arbitrary point inside the 3D grid and bins macro-particles by using 3x3x3 points interpolation scheme.

The Grid3D class has methods that distribute the 3D grid over processors. For parallel calculations each CPU keeps its Grid3D object but all of them together can be considered as one big distributed 3D grid. In this case each separate Grid3D object has two additional 2D slices that have the same values as these slices on neighboring CPUs to provide identical results for 3x3x3 points interpolation schema for parallel and nonparallel calculations.

Surface Classes

The structure of the surface classes is shown in Fig. 3. Each surface class is a subclass of an abstract baseSurface class and each must implement at least one method: "impact". This method removes the macro-electron with a particular index from the eBunch and adds the emitted new macro-electrons to the eBunch at the specific point on the beam-pipe surface and with a momentum defined relative to the normal vector to the surface.

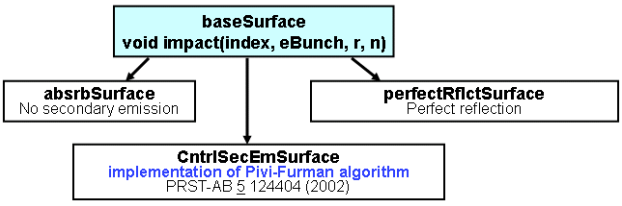

Figure 3: The hierarchy of surface classes

At this moment the EC module has three surface classes:

- An absorption surface that absorbs all electrons.
- A perfect reflection surface.
- A Controlled Secondary Emission surface that is an implementation of Pivi-Furman's algorithm [4].

The original Pivi-Furman algorithm has been modified to enable a variable macro-size for macro-electrons. This modification enables the user to control the number of macro-electrons during costly electron cloud buildup simulations.

Field Source Classes

The field source classes specify magnetic and electric fields that act on electrons moving inside electron cloud region. All these classes implement an interface that is defined by the parent class of this hierarchy. Only two methods should be implemented: "get electric field" and "get magnetic field". They use coordinates as input parameters and return the three components of the fields at the input coordinates. At this moment, the EC module includes three child classes of the base field source: the electric field of the electron cloud, electric and magnetic fields from the proton bunch, and a source of uniform electric and magnetic fields. Further options are under development.

The field source class structure allows the creation of new classes for more complicated magnetic and electric fields.

Tracker Classes

The tracker classes move macro-electrons in the EC region. There is a parent class (baseParticleTracker) that specifies the interface and functionality of all trackers. All subclasses should implement only one method. Inside this method macro-electrons are moved by using the combined forces of all registered electro-magnetic field sources. Users can register as many field sources as they wish.

At present there are two methods implemented to calculate the non-relativistic electron's motion:

- Symplectic integration using a leapfrog method
- Analytic integration using a constant local field approximation.

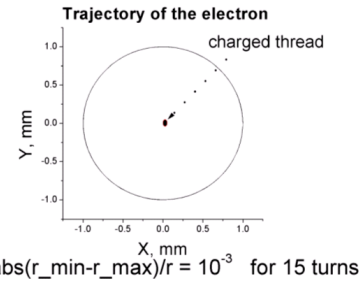

Figure 4: Circular electron motion.

BENCHMARKS WITH ANALITICAL MODELS

A set of benchmarks has been performed to verify the fidelity of different parts of the EC module. The most sophisticated combine benchmarks of the space charge solver and the tracker. Figure 4 shows the two-dimensional motion of an electron in the electric field of a thin charged wire that is perpendicular to the plane of

motion. The correct circular motion is reproduced. In other benchmarks the ability of the tracker to calculate a trajectory in the presence of magnetic fields has also been checked.

SIMULATION ALGORITHMS

It was pointed out above that only two classes connect the electron cloud module to the rest of the ORBIT code and orchestrate all simulations (see Fig. 2).

The EP_Node class is a subclass of the Node class of ORBIT:

- It represents an accelerator lattice element through which the proton bunch is propagated.
- It uses an EP_NodeCalculator to propagate the proton bunch through the electron cloud region.
- There can be an arbitrary number of these nodes in the lattice.

EP_NodeCalculaor is the class that actually combines all classes together and implements the algorithm of EC calculations. The algorithm can be divided onto three stages:

The first stage includes preparations for calculation: checking the sizes of the arrays and resizing if necessary; analyzing the proton bunch analysis; and calculating the fields.

In the second stage, the simulation of the electron cloud buildup is performed and its effect on the proton bunch is accumulated.

In the third stage, the accumulated kicks are applied to the protons of accelerator beam.

Stage 1. Preparations

This stage deals with the proton bunch only. The macro-particles of this bunch are distributed among CPUs. This task is carried out by the ParticleDistributor ORBIT class. By using the information from this class the necessary 3D arrays are resized on each CPU. It is necessary to have 5 distributed 3D grid objects: two for space charge proton density and potential and three for x, y, and z directions of accumulated kicks. The macro-protons from the bunch are binned into the space charge density grid and the space charge potential is calculated. The application of boundary conditions on the perfect conducting beam pipe surface is optional.

Stage 2. EC Buildup Simulation

During this stage, as time progresses, the proton grid is moved through the electron cloud region at the beam velocity and the processes of primary electron generation, macro-electron motion, and multipaction are simulated. This is done by three nested loops and is a typical technique for EC buildup algorithms.

The outside nested loop prepares the EC potential as a field source for EC dynamics simulation. The number of steps (usually a few thousand) is determined by the requirement of adiabatic change in the electron cloud potential. In the beginning of this iteration, primary electrons are generated by routines simulating protons grazing the vacuum chamber or ionizing residual gas. The generated macro-electrons are distributed randomly between CPUs. During the calculations they reside at the same CPU where they have been generated. No further macro-electron distributor is needed. Next the space charge potential of the electrons is calculated. This potential is the sum of all potentials over all CPUs, so communications between CPUs are needed. This potential is used as one of the field sources for the Tracker. Also before the end of this iteration the momentum kicks to the proton bunch are accumulated.

During the intermediate nested loop the potential of proton beam at each specific position is used to update the proton beam field source for the Tracker. This step requires communication between CPUs because the p-beam potential grid is distributed. This loop is necessary because the adiabatic changes in the proton beam potential may be important, so there it must be possible to update the proton beam field source as frequently as necessary. Usually it is sufficient to update fields simultaneously, which means this loop needs only one step.

Inside the third loop the electron motion is integrated and the Pivi-Furman electron-wall interaction model is applied to those that impact the wall.

Stage 3. Proton Coordinate Updates

This stage applies the accumulated kicks from electrons to the protons.

CONCLUSION

The Electron Cloud Module is now fully implemented in the ORBIT code and enables self consistent ECE simulations.

REFERENCES

[1] R. J. Macek, A. A. Browman, M. J. Borden, D. H. Fitzgerald, R. C. McCrady, T. Spickermannn, and T. J. Zaugg, "Experimental Studies of Electron Cloud Effects at the Los Alamos PSR: a Status Report", Proc. ECLOUD'04, Napa, April 2004; to be published.

[2] H.C. Hseuh, "Design and Implementation of SNS Accumulator Ring Vacuum System with Suppression of Electron Cloud Instability", Proc. ECLOUD'04, Napa, April 2004; to be published.

[3] J. A. Holmes, V. Danilov, J. Galambos, A. Shishlo, S. Cousineau, W. Chou, L. Michelotti, F. Ostiguy, J. Wei, "ORBIT: beam dynamics calculations for high-intensity rings", Proc. EPAC'02, Paris, June 2002, p. 1022.

[4] M. T. F. Pivi and M. A. Furman, PRST-AB **6** (2003) 034201.